TU Clausthal

Clausthal University of Technology

Continuous Edge Gradient-Based Template Matching for Articulated Objects

Daniel Mohr and Gabriel Zachmann

IfI Technical Report Series **IfI-09-01**

Department of Informatics
Clausthal University of Technology

Impressum

Publisher: Institut für Informatik, Technische Universität Clausthal
Julius-Albert Str. 4, 38678 Clausthal-Zellerfeld, Germany
Editor of the series: Jürgen Dix
Technical editor: Wojciech Jamroga
Contact: wjamroga@in.tu-clausthal.de
URL: http://www.in.tu-clausthal.de/forschung/technical-reports/
ISSN: 1860-8477

The IfI Review Board

Continuous Edge Gradient-Based Template Matching for Articulated Objects

Daniel Mohr and Gabriel Zachmann

Abstract

Detection of articulated objects in images, including location and state, is an important and challenging task in many object tracking applications. Image edges have proven to be a key feature, although their quality is influenced by many factors.
In this paper, we propose a novel edge gradient-based template matching method for object detection. In contrast to other methods, ours does not perform any binarization or discretization during the online matching. This is facilitated by a new continuous edge gradient similarity measure. Its main components are a novel edge gradient operator, which is applied to query and template images, and the formulation as a convolution, which can be computed very efficiently in Fourier space.
Our method consists of a preprocessing stage for the template images, a simple preprocessing of the query image, and our similarity measure between template and query image, which yields a confidence map for the query image. The resulting confidence map can be used for the final object localization.
We compared our method to a state-of-the-art chamfer-based matching method. The results demonstrate that our method is much more robust against weak edge response and yields much better confidence maps with fewer maxima that are also more significant. In addition, our method lends itself well to efficient implementation on GPUs: at a query image resolution of 320×256 and a template resolution of 80×80 we can generate about 330 confidence maps per second.

1 INTRODUCTION

Detection and tracking of articulated objects is used in many computer vision applications, such as tracking people, computer-assisted car steering, or human-computer interaction. Our long-term goal is precise tracking of the human hand in order to be able to use it as a dextrous input device for virtual environments and many other applications.

An important initial step in object tracking is to localize the object in the 2D image delivered by the camera. This is a challenging task especially with articulated objects, due to the huge state space and, possibly, time constraints. Most approaches formulate tracking of articulated objects as detecting multiple object: given a database of many objects, find the object from the database that best matches the object shown in the input image, which also involves finding the location in the input image where that best match

occurs. Each of the objects in the database represents the articulated object in a different state and viewpoint. Typically, the database consists of images, called *templates*, which are possibly preprocessed. This can result in a database size of thousands of templates. Thus, tracking of articulated objects can be reformulated as template matching. Of course, this approach can also be applied to detection/tracking of rigid objects. For rigid objects, the number of templates is determined only by the different orientations and scales.

A crucial premise are good features that allow to distinguish between the target object and the background and between different states of the object. The shape of most articulated objects is very characteristic and does not appear in the background. Therefore, a powerful method to match templates is to compare the edge images of template and input image. However, the quality of the edge images is negatively influenced by various factors such as scene illumination, camera parameters, object and background color, shadows, etc.

In this paper, we propose a novel method for template matching based on edge features, which is robust against varying edge response. To this end, we propose a novel similarity measure between a template and the query image that utilizes the continuous edge gradient (orientation *and* intensity). The input to our algorithm is a query image and a set of templates. The output is a *confidence map*, storing for each position in the query image the index and similarity of the best matching template (note that, usually, the best matching template is different for each position).

In subsequent steps, this confidence map can be used directly to extract the best match, or it can be combined with other confidence maps using different features. This is, however, not our focus here.

Main Contributions: Our method does not perform any binarization or discretization during the online matching process. By contrast, all current methods based on edge distance/similarity need binary edge images. This incurs thresholds that are difficult to adjust automatically, which reduces the robustness of these approaches.

We utilize the orientation and intensity of edges of both the templates and the query images directly in our similarity measure. By contrast, most current methods discretize edge orientations into a few intervals, which renders the similarity measure discontinuous with respect to rotation of the object.

Our method is well suited for a complete implementation in the stream processing model (e.g., on modern GPUs), which allows for extremely fast template matching.

2 RELATED WORK

The most often used edge based approaches to template matching compare binarized edge images: the dissimilarity between two edge images I_A and I_B is defined as the distance between the edge pixel set A of I_A and edge pixel set B of I_B. The directed chamfer distance [Barrow et al., 1977],[Borgefors, 1988] $\mathcal{C}$ from set A to B with re-

spect to metric d is defined as

$$\mathcal{C}(A,B) = \frac{1}{|A|} \sum_{a_i \in A} \min_{b_j \in B} d(a_i, b_j) \tag{1}$$

The chamfer distance can be formulated as a convolution of image I_A with the distance transform of image I_B. A disadvantage of the chamfer distance is its sensitivity to outliers. Chamfer matching for tracking of articulated objects is, for example, used by [Stenger et al., 2006], [Athitsos and Sclaroff, 2001], [Athitsos and Sclaroff, 2002], [Athitsos et al., 2004], [Gavrila and Philomin, 1999], [Sudderth et al., 2004], [Kato et al., 2006] and [Lin et al., 2007].

Another distance measure is the Hausdorff distance. The directed Hausdorff distance [Huttenlocher et al., 1993] is defined as the maximum of all distances from each point in A to its nearest neighbor in B:

$$\mathcal{H}(A,B) = \max_{a_i \in A} \{ \min_{b_j \in B} \{ d(a_i, b_j \} \}. \tag{2}$$

The generalized form uses the $k\mathrm{th}$ largest distance instead of the maximum,

$$\mathcal{H}(A,B) = \underset{a_i \in A}{k\mathrm{th}} \{ \min_{b_j \in B} \{ d(a_i, b_j \} \} \tag{3}$$

where $k\mathrm{th}$ returns the k-largest value. The value k can be used to control the number of outliers that are tolerated.

Both, chamfer and Hausdorff distance can be modified to take edge orientation into account with limited accuracy. One way to do this is to split the template and query images into b separate images, each containing only edge pixels within a predefined orientation interval [Thayananthan et al., 2006], [Stenger et al., 2006]. To achieve some robustness against outliers, [Stenger et al., 2006] additionally limited the nearest neighbor distance from a point of set A to set B by a predefined upper bound. A disadvantage of these approaches is, of course, the discretization of the edge orientations, which can cause wrong edge distance estimations.

[Olson and Huttenlocher, 1997] integrated edge orientation into the Hausdorff distance. They modeled each pixel as a 3D-vector. The first two components contain the pixel coordinates, the third component the edge orientation. The maximum norm is used to calculate the pixel-to-pixel distance. [Sudderth et al., 2004] presented a similar approach to incorporate edge position and orientation into chamfer distances. If their approach is used to detect the location of an object in the image, it has to be implemented directly, because it cannot be formulated as convolution. This results in high computation times. Edge orientation information is also used by [Shaknarovich et al., 2003] as a distance measure between templates. They discretized the orientation into four intervals and generate an orientation histogram. Because they do not take the edge intensity into account, the weight of edge orientations resulting from noise is equal to that of object edges, which results in a very noise sensitive algorithm.

In [Athitsos and Sclaroff, 2003] the templates are stored as line vectors, each line containing center position, orientation, and length. The line extraction thresholds are

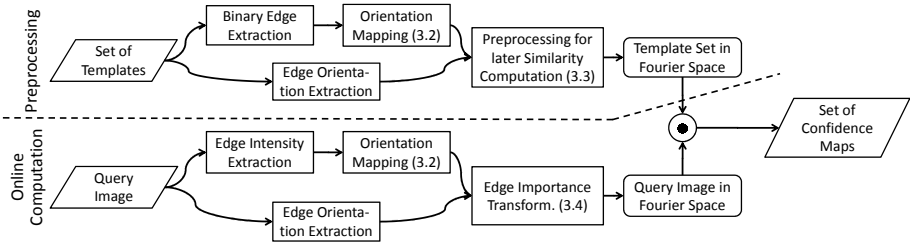

Figure 1: Overview of our approach. The numbers in parentheses denote the section describing the respective stage.

set such that most lines belonging to the target object are found. This results in very low thresholds, which has the disadvantage that many edges caused by noise are extracted, too. Consequently, the image becomes highly cluttered. Edges are combined to straight lines. Matching is formulated as finding the best correspondences between template and input lines. Because a large number of edges, produced by noise, are processed in the line matching step, the probability of false matching is highly dependent on the input image quality and background.

3 The Continuous Edge Image Similarity Measure

Before describing our approach, we introduce the following notation:

$\mathcal{T} = \{T_k\}$ is a set of templates with $k = 0 \cdots l-1$,
$W_k \times H_k$ is the size of T_k,
E_{T_k} is the binarized edge image of T_k,
I_Q a query image of size $W_Q \times H_Q$,
$I_Q^{\mathbf{c},k} \subset I_Q$ a sub-image of size $W_k \times H_k$ and centered at $\mathbf{c} \in [0, W_Q] \times [0, H_Q]$,
E_Q the edge intensity image of I_Q, and
$\mathcal{S}_{I_Q}(k, \mathbf{c})$ a similarity measure between $I_Q^{\mathbf{c},k}$ and T_k with the co-domain $[0, 1]$, in the sense that the value 1 indicates a perfect match.

The goal of a template matching algorithm is to find the template index $\bar{k}$ that is most similar to the target object in the image and its correct image coordinates $\bar{\mathbf{c}}$. This can be achieved in two steps: first, calculate the image similarities for some or all $\mathbf{c}$ and k, and, second, based on the similarities, obtained in step 1, choose an appropriate $\bar{k}$ and $\bar{\mathbf{c}}$ to represent the object state and position. The latter is not the focus of this paper.

Due to loss of information (2D projection, camera noise, etc.), salient features are needed to get results of high quality. Edges are such a feature, which are fairly robust against illumination changes and varying object color, and because they also work with untextured objects. However, edges are not completely independent from illumination, color, texture, and camera parameters. Therefore, a robust algorithm for efficient template matching is needed.

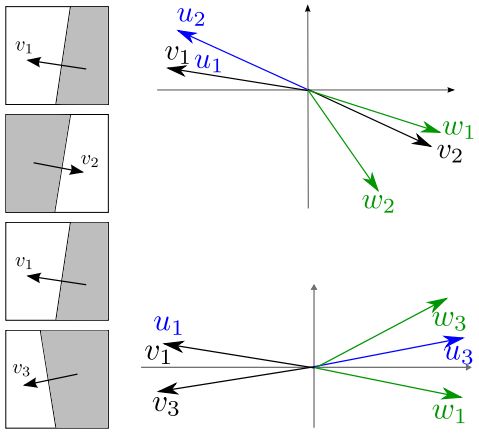

Figure 2: In order to achieve a consistent gradient distance, we map gradients as shown here, before actually comparing them. That way, our edge similarity measure returns low "distances" for the edge gradients in both situations shown here. The v_i denote the original gradients, u_i are intermediate ones, and w_i are the final ones that are further used.

3.1 Overview of our Approach

Our approach consists of two stages. First, the template set $\mathcal{T}$ and the query image I_Q are preprocessed to allow efficient edge-based template matching; second, the matching itself is performed, which computes a similarity value for all templates T_k and all sub-images $I_Q^{\mathbf{c},k}$ for all query image pixels $\mathbf{c}$.

The templates are preprocessed in two steps. First, we generate images of the object in different states and viewpoints. An edge extractor is used to obtain a binary edge image. Then, we extract the edge gradient at the edge pixels. This gradient is then mapped in a way so that they can be compared easily and correctly (see Section 3.2). Second, we transform the template image such that the similarity between template and query image can be calculated efficiently by a convolution (Section 3.3).

Before computing similarities, we extract the edge intensities and gradients from the query image and map them, just like the preprocessing for the templates. In order to overcome the problem of multiple edges, caused by noise, shadows, and other effects, we further transform the image appropriately (Section 3.4).

3.2 Consistent Gradient Distance

Depending on background color and illumination, the edge gradient points into the foreground or away from it (see Figure 2). Thus, edges whose orientations differ exactly by π need to be treated as identical. Taking this into account, the similarity between two gradient vectors $\mathbf{v_1}$ and $\mathbf{v_2}$ *could* be simply calculated by $|\mathbf{v_1} \cdot \mathbf{v_2}|$. However, later, we want to express this similarity as a convolution operator, but the absolute value is non-linear and, consequently, cannot be performed in Fourier space. This would greatly

increase the computation time (details are described in Section 3.5).

Therefore, we propose to map the gradient vectors such that the mapped gradient can be used in a similarity measure in Fourier space. We define our mapping function $\hat{\mathbf{v}} = f(\mathbf{v})$ as follows:

$$\theta = \arctan \frac{v_y}{v_x} \tag{4}$$

$$\theta' = \begin{cases} \theta & \theta \geq 0 \\ \theta + \pi & \theta < 0 \end{cases} \tag{5}$$

$$\hat{\mathbf{v}} = (\hat{v}_x, \hat{v}_y) = \|\mathbf{v}\|_2 \cdot (\cos(2\theta'), \sin(2\theta')) \tag{6}$$

Now, we can calculate the similarity between $\hat{\mathbf{v}}_\mathbf{1}$ and $\hat{\mathbf{v}}_\mathbf{2}$ simply by $\hat{\mathbf{v}}_\mathbf{1} \cdot \hat{\mathbf{v}}_\mathbf{2}$. Figure 2 illustrates the problem and our mapping.

To achieve higher robustness, we avoid to apply any kind of edge binarization algorithm to the input image, because this would introduce at least a threshold parameter, which is always difficult to adjust. Instead, we interpret the edge intensity values of the input image as probabilities of the corresponding pixel to be an edge.

In the next section, we develop an algorithm that calculates an edge similarity that utilizes these probabilities directly. By contrast, common approaches like chamfer or Hausdorff matching need a binarized input image.

3.3 Computing the Similarity of Edge Images

In this section, we describe the core of our approach, the matching of a template T_k and a query image I_Q. We assume we are given the following information:

$L_{T_k} = \{\mathbf{c} \mid E_{T_k}(\mathbf{c}) = 1\}$	the edge pixel list;
$\hat{G}_T$ and $\hat{G}_Q$	the mapped edge gradients of the template and query image, resp., additionally with each gradient vector normalized to length one;
$\mathcal{N}(\mathbf{x})$	the pixel neighborhood of $\mathbf{x}$ with size $n \times n$;
$K : \mathbb{R} \to [0, 1]$	a unimodal function (kernel function) with the maximum at $K(0) = 1$; and

A possible choice for K is the Gaussian function. In the following we will use a kernel function with bounded support:

$$\tilde{K}(\mathbf{x}, h, n) = \begin{cases} K(\frac{\|\mathbf{x}\|_2}{h}) & \|\mathbf{x}\|_\infty \leq n \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

As explained previously, we do not have a discrete set of edge pixels in the query image, and, thus, cannot calculate directly a distance from each edge pixel $\mathbf{e} \in L_{T_k}$ to the closest edge pixel in I_Q. Instead, we use probabilities to estimate the distance: the

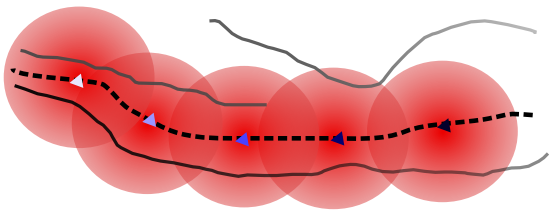

Figure 3: We estimate the similarity between a template edge (dashed line) and a query image edge, which is represented by intensities (gray solid curves), by multiplying a kernel that is centered around each template edge pixel (circles) with the edge intensities. The intensity value of the template edge pixel (triangles) visualize the "closeness" of query image edges.

higher the probability and the nearer a pixel in the query image is to a template edge pixel, the lower the distance should be. The mean probability of a neighborhood of $\mathbf{e}$ is used as inverse distance measure, so that a small distance results in a high mean value and vice versa. The weight of the neighboring pixels is controlled by the choice of the kernel function K and its parameter h. Because only close pixels are relevant for the similarity measure, we only take into account a neighborhood of each template edge pixel of size $n \in \mathbb{N}$.

To compute the similarity $\mathcal{S}_{I_Q}(k, \mathbf{c})$, we calculate for each edge pixel $\mathbf{e}$ in the template image the probability $P^{\mathbf{c},k}(\mathbf{e})$ that an edge in the query image is close to it:

$$P^{\mathbf{c},k}(\mathbf{e}) = \frac{1}{2} + \frac{1}{C_K} \sum_{\mathbf{p} \in \mathcal{N}(\mathbf{e})} \tilde{K}(\mathbf{p} - \mathbf{e}, h, n)\, E_Q(\mathbf{c} + \mathbf{p}) \hat{G}_T(\mathbf{e}) \cdot \hat{G}_Q(\mathbf{c} + \mathbf{p}) \tag{8}$$

with the normalization factor

$$C_K = 2 \cdot \sum_{\mathbf{p} \in \mathcal{N}(\mathbf{e})} \tilde{K}(\mathbf{p} - \mathbf{e}, h, n) \tag{9}$$

Note that $\hat{G}_T(\mathbf{e}) \cdot \hat{G}_Q(\mathbf{c} + \mathbf{p})$ is a 2D scalar product; because this is in $[-1, 1]$, we have to use an offset. Figure 3 illustrates the idea behind this measure.

Then, we define the overall similarity as the mean probability

$$\mathcal{S}_{I_Q}(k, \mathbf{c}) = \frac{1}{|L_{T_k}|} \sum_{e \in L_{T_k}} P^{\mathbf{c},k}(\mathbf{e}) \tag{10}$$

Since the kernel function K and parameters h and n are fixed, the normalization factor C_K is constant. Therefore, we can rewrite Eq. 8 as

$$P^{\mathbf{c},k}(\mathbf{e}) = \frac{1}{2} + \sum_{\mathbf{p} \in \mathcal{N}(\mathbf{e})} \eta_T(\mathbf{p}, \mathbf{e}) \cdot \eta_Q(\mathbf{c} + \mathbf{p}) \tag{11}$$

with

$$\eta_T(\mathbf{p}, \mathbf{e}) = \frac{1}{C_K} \tilde{K}\left(\mathbf{p} - \mathbf{e}, h, n\right) \hat{G}_T(\mathbf{e}) \tag{12}$$

$$\eta_Q(\mathbf{x}) = E_Q(\mathbf{x})\, \hat{G}_Q(\mathbf{x}) \tag{13}$$

$$\tag{14}$$

and insert it into Eq. 10

$$\mathcal{S}_{I_Q}(k, \mathbf{c}) = \frac{1}{2} + \frac{1}{|L_{T_k}|} \sum_{e \in L_{T_k}} \sum_{\mathbf{p} \in \mathcal{N}(\mathbf{e})} \eta_T(\mathbf{p}, \mathbf{e}) \eta_Q(\mathbf{c} + \mathbf{p}) \tag{15}$$

Because $\tilde{K}$ is zero everywhere outside its support, we can rewrite the inner sum as a sum over all pixels in T_k. Similarly, the outer sum can be rewritten, yielding

$$\frac{1}{2} + \frac{1}{|L_{T_k}|} \sum_{\mathbf{y} \in D_k} \left(E_{T_k}(\mathbf{y}) \sum_{\mathbf{x} \in D_k} \eta_T(\mathbf{x}, \mathbf{y}) \eta_Q(\mathbf{c} + \mathbf{x}) \right) \tag{16}$$

where $D_k = [0, W_k] \times [0, H_k]$. We rewrite again:

$$\frac{1}{2} + \sum_{\mathbf{x} \in D_k} \Big(\eta_Q(\mathbf{c}, \mathbf{x}) \underbrace{\frac{1}{|L_{T_k}|} \sum_{\mathbf{y} \in D_k} E_{T_k}(\mathbf{y}) \eta_T(\mathbf{x}, \mathbf{y})}_{\tilde{E}_{T_k}(\mathbf{x})} \Big) \tag{17}$$

Notice that $\tilde{E}_{T_k}$ can be calculated offline. Finally, we arrive at

$$\mathcal{S}_{I_Q}(k, \mathbf{c}) = \frac{1}{2} + \sum_{\mathbf{x} \in D_k} \eta_Q(\mathbf{c} + \mathbf{x}) \cdot \tilde{E}_{T_k}(\mathbf{x}). \tag{18}$$

$\mathcal{S}_{I_Q}(k)$ is called the *confidence map* of I_Q and T_k and is basically generated by correlating $\tilde{E}_{T_k}$ with $E_Q \hat{G}_Q$ (see Eq. 11). It can be calculated efficiently in Fourier space by expressing the correlation as a convolution by flipping the image $\tilde{E}_{T_k}$. Since $\eta_T, \eta_Q \in \mathbb{R}^2$, we compute Eq. 18 independently for each component, x and y, so that they are scalar-valued correlations.

So far, we have described a robust and fast method to compute the edge similarity between a query image and a set of templates. One remaining problem is that a query image often contains multiple edges close to each other, which are, therefore, also close to the appropriate template edge. For instance, a cable, which produces a shadow, causes four instead of two strong edges. Depending on the edge orientation, this causes severe over- or underestimation of $P^{\mathbf{c},k}(\mathbf{e})$.

The next section describes our method to overcome this problem by preprocessing the query image. Note that this will not increase the computation time significantly (see Section 4).

3.4 Preprocessing the Query Image

In the following, we assume that the query image contains intensities only, and that the edge gradient for each pixel is given.

It is obvious that the larger the intensity of an edge pixel is, the higher its weight should be in the similarity measure defined in Section 3.3. This could easily be incorporated into Eq. 8 by normalizing it with the query image neighborhood. However, this would have the disadvantage that the lower the number of significant edges, the lower the signal to noise ratio would be. In the extreme case of a region that contains no useful edges, the measure matches only to noise and, thus, has no significance.

Therefore, we propose an approach that does not exhibit this problem: we preprocess the query image, such that at each pixel, the intensity weighted edge information of the neighborhood is stored. Note that intensity values and gradients of the neighborhood are combined in different ways, which are explained in the following.

The new edge gradient at each pixel is computed as the weighted average gradient of its neighborhood:

$$\tilde{G}_Q(\mathbf{x}) = \frac{f(\mathbf{x})}{\|f(\mathbf{x})\|_2} \tag{19}$$

with

$$f(\mathbf{x}) = \sum_{\mathbf{p}\in\mathcal{N}(\mathbf{x})} K\left(\frac{\mathbf{x}-\mathbf{p}}{h}\right) I_Q(\mathbf{p})\hat{G}_Q(\mathbf{p}). \tag{20}$$

Thus, the higher the edge intensity at a pixel is, the more important its orientation information is.

In contrast to orientations, intensities cannot be averaged, because in regions with many strong edges, for example caused by shadows, we would get an unrealistically high new intensity. Instead, keeping just the intensity of the strongest neighboring edge, weighted by distance, is a much better choice. This is realized by the following function:

$$I_Q(\mathbf{x}) = \max_{\mathbf{p}\in\mathcal{N}(\mathbf{x})} K\left(\frac{\mathbf{x}-\mathbf{p}}{h}\right) I_Q(\mathbf{p}) \tag{21}$$

Figure 4 shows by way of an example that this yields the desired result, in contrast to the weighted average.

3.5 Implementation

Our method lends itself well to implementation on modern GPUs using the stream programming model, which we describe in this section using the CUDA programming environment [Nvidia, 2008]. Since the convolution kernel size in both Eqs. 20 and 21 is fairly small, it is more efficient to implement the complete query image preprocessing directly in a single CUDA kernel,[1] without FFT.

[1] In the context of stream processing, the term "kernel" denotes a function that is applied to each item in a stream (i.e., a homogenous array).

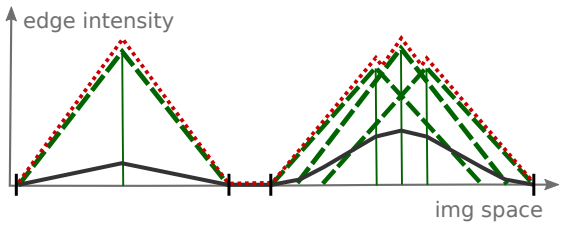

Figure 4: Our query image preprocessing takes the maximum (dotted line) of the weighted kernel functions (dashed lines) in the neighborhood. This is a much better choice than the weighted average (solid curve). Notice that the $\max$ preprocessing yields a response of the same magnitude in the left and the right example, whereas the average produces undesirable varying responses.

We load the image into a 2D texture. For each output pixel, the CUDA kernel executes a loop over the $(2n+1) \times (2n+1)$ neighboring pixels performing the algorithms described in Sections 3.4. The confidence map $\mathcal{S}_{I_Q}(k)$ is calculated componentwise for the x and y direction (see Section 3.3), and the results are summed up. Calculating each component (x, y) of $\mathcal{S}$ can be formulated as a linear convolution. The kernel size is equivalent to the size of the template images (see Eq. 18), and thus relatively large. To accelerate the confidence map generation, one should perform the convolution through multiplication in Fourier space.

$$\mathcal{S}_{I_Q} = F^{-1}(F(E_Q \hat{G}^x_Q) F(\text{flip}(\tilde{E}^x_{T_K})) + F(E_Q \hat{G}^y_Q) F(\text{flip}(\tilde{E}^y_{T_K}))) \tag{22}$$

F and F^{-1} denote the Fourier transform (FT) and the inverse FT, resp. Because of the linearity of the (inverse) Fourier transform

$$F^{-1}(F(I^x_Q * I^x_T + I^y_Q * I^y_T)) =$$
$$F^{-1}(F(I^x_Q * I^x_T) + F(I^y_Q * I^y_T))$$

we can accumulate the results of the convolution in Fourier space and save one inverse FT. In the above formula, $*$ denotes the convolution operator.

The template images are preprocessed and Fourier transformed offline. Since with many object localization applications, a lot of localizations are performed with the same set of templates, it makes sense to upload all Fourier transformed templates to the GPU memory. This greatly improves memory access speed during the computation of the confidence map. In our implementation, we use the FFT library from NVIDIA.

Due to the high number of templates and the limited number of memory on the grapics hardware, the memory consumption of each fourier transformed template is an important factor. In the following section, we will discuss, how the memory usage of the fourier transformed templates could heavily be reduced.

3.6 Efficient Template Storage

The fourier transformation of query and template images have to be performed in a common spatial domain. Its width/height is the sum of the query and template image width/height, rounded up to the next power of two for most efficient calculation. Therefore, storing the fourier transformed template consumes a huge amount of memory. However, by construction our templates contain only low frequencies (Section 3.3). The idea to skip the higher frequences of the fourier transform is straightforward. The remaining question is how many of the higher frequencies can be skipped without losing significant information in the template while saving as much memory as possible.

Crucial for further object detection algorithms are the confidence maps, generated by convolving a query image with a template. We can compress the template images as long as the confidence map, generated by the compressed template, here denoted with $\mathcal{S}^f_{I_Q}$, does not differ too much from the exact confidence map $\mathcal{S}_{I_Q}$, generated by the uncompressed template $\tilde{E}_T$. More specifically, for a set of templates and the query image at each position in the query image, the best matching template index and its similarity value is of further interest. This information is given by the *combined confidence map* defined as

$$\mathcal{S}_{I_Q}(x,y) = \max_{k\in[0,l-1]} \left\{\mathcal{S}_{I_Q}(k,x,y)\right\}. \tag{23}$$

Let $F_T = F(\tilde{E}_T)$ be the Fourier transform of a preprocessed template image $\tilde{E}_T$ and F^f_T the fourier transformed template, containing only f percent of the frequences in x- and y-direction stored in F_T. The storage cost of F^f_T is by the factor $(f/100)^2$ lower compared to F_T. The error we make when using F^f_T instead of F_T to generate the combined confidence map between a query image I_Q and a set of templates $\mathcal{T}$, is defined as the RMS (root mean square) error between the exact combined confidence map $\mathcal{S}_{I_Q}$, based on F_T and the combined confidence map $\mathcal{S}^f_{I_Q}$ based on F^f_T:

$$E_{I_Q}(f) = \sqrt{\frac{1}{W_Q H_Q}\sum_{x,y}\left(\mathcal{S}_{I_Q}(x,y) - \mathcal{S}^f_{I_Q}(x,y)\right)^2} \tag{24}$$

To become as independent as possible from the query image I_Q we use the average RMS error of a large set of query images $\{I^j_Q | j = 1 \ldots N\}$:

$$E(f) = \frac{1}{N}\sum_{i=1}^{N} E_{I^i_Q}(f) \tag{25}$$

We captured 8 different image sequences (four sequences with different background and amount of edges, each of them captured under two different lighting conditions) with a total of about one thousand frames. Each sequence contains images showing pointing hand, open hand, and open-close gestures as well as images without a valid hand pose. Figure 5 shows the plot of the average RMS error for different f. We have decided to set $f = 19/12/15$ for template dataset 1/2/3 respectively.

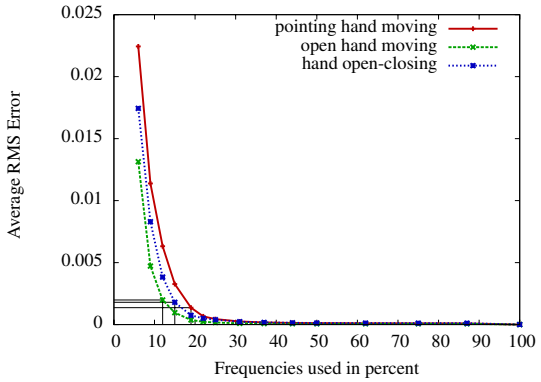

Figure 5: For each template datase: the average root mean square error of combined confidence maps built using templates with only the lower f percent of the image frequencies. The error is measured relative to the combined confidence map using the uncompressed templates (all frequencies).

4 Results

In our datasets, we use the human hand as the object to be detected. In the field of articulated object detection and tracking, the chamfer and Hausdorff distance measures are most often used as distance measure for edge images. Stenger showed in [Stenger, 2004] that the chamfer outperforms the Hausdorff matching for human hand templates. Therefore, we compare our method with the chamfer matching algorithm.

For comparison, we need an appropriate measure for the ability of the methods to localize an object at the correct position in the query image. Given a query image I_Q, both the chamfer and our method generate a confidence map $\mathcal{S}_{I_Q}(k)$ for each template T_k. Now let $(\hat{x}, \hat{y})$ be the true location of the object in the query image and $\hat{c}$ the matching value at $(\hat{x}, \hat{y})$ of the template, delivering the best match according to the approach used. Obviously, the fewer values in all confidence maps are better than $\hat{c}$, the better the matching algorithm is.[2] This is the idea of our quality measure of the matching algorithms. Our quality measure is an indicator for the number of other matching values in S_{I_Q} that are higher or lower. The higher the quality measure of the approach is, the more matching values at other then the correct position are lower and thus the better the template matching approach is. The chamfer matching, of course, returns distances, not similarity values, but the chamfer matching output can be converted easily into similarity values by inverting them.

[2] The chamfer matching, of course, returns distances, not similarity values, but the chamfer matching output can be converted easily into similarity values by inverting them.

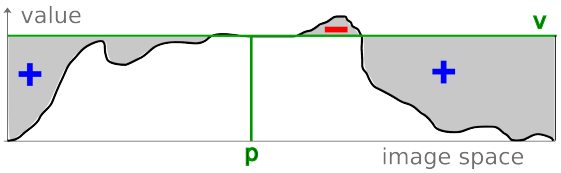

Figure 6: 1D example of our quality measure: the true location p of the target object is determined manually, v is the value at p in the combined confidence map. Our quality measure is basically the sum of the signed gray areas over the whole confidence map.

In detail, we use the following quality measure:

$$\mathcal{Q}_{I_Q} = \frac{1}{N} \sum_{\substack{0 \le x < W_Q \\ 0 \le y < W_H}} \left(\mathcal{S}_{I_Q}(\hat{x}, \hat{y}) - \mathcal{S}_{I_Q}(x, y) \right) \tag{26}$$

with

$$N = W_Q H_Q (\max - \min) \tag{27}$$

where min and max are the smallest and largest value in $\mathcal{S}_{I_Q}$, resp. We manually determined the true object positions $(\hat{x}, \hat{y})$. Thus, the higher the value $\mathcal{Q}_{I_Q}$, the better the method works for the query image and template set. Figure 6 illustrates the measure by means of a 1D combined confidence map.

Of course, a better quality measure would also take into account the index of the true template. Unfortunately, we did not yet have the time to manually label the templates. But, we observed that at the true position, the best matching template reported by our algorithm looks very similar to the object in the input image in most frames. Video sequences, demonstrating this observation, can be found at `http://cg.in.tu-clausthal.de/research/handtracking/index.shtml`.

As test data we used RGB images of resolution 320×256. We converted them to gray scale, then applied a Gaussian filter of size 3×3 to reduce noise, the Sobel filter to extract the edge gradient, and finally a non-maximum suppression filter. The resulting images are then transformed as explained in Section 3.4. All preprocessing is done on the graphics hardware in CUDA.

Some query images contain edge response values differing strongly from the average, for example, a very bright object in front of a black background. To overcome this problem, the logarithm can be applied to the edge intensities. We have found that, in practice, some scenarios work better with, some without this modification. The main reason is that the edge noise is often intensified.

We used three datasets for evaluation (see Figures 8, 9, 10). Dataset 1, consisting of 200 frames, is a pointing hand moving in the image. The templates (see Figure 7) are 300 renderings of an artificial 3D hand model representing a pointing gesture. Each template is generated from a different camera viewpoint. In dataset 2, an open hand

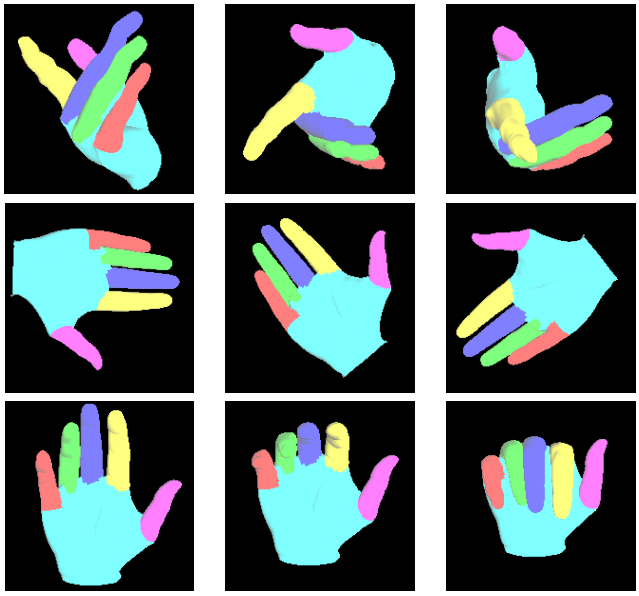

Figure 7: Each row shows three examples of our rendered hand, which we used to generate the templates. (pointing hand in the 1st row, open hand on the 2nd one and open-closing hand in the 3rd one

is tested. The length of the dataset is 200 frames, too, and the number of templates is 300 as well. Dataset 3 shows an open-closing sequence of a human hand, consisting of 135 frames. Again, the templates are created using the 3D hand model, with its fingers opening and closing, rendered from three different camera angles.

The template edges are extracted through the Canny edge detector from the depth buffer of the renderings. Here, we have well-known conditions, so we can manually optimize thresholds for the Canny detector. As kernel function we have chosen the Gaussian function $K(x) = e^{-\frac{1}{2}x^2}$. The bandwidth parameter h, needed in Eq. 7, has been manually optimized; it depends only on the templates, not on the query images. We set $n = \lceil 3h \rceil$ (three sigma rule for Gaussians) and $h = 3.3$ for dataset 1 and $h = 4.0$ for datasets 2 and 3. The resolution of the template images depends on the object shape, distance, and orientation relative to the camera. For our experiments we have an average template size of 80×80.

For the chamfer based template matching algorithm we used the parameters proposed by [Stenger et al., 2006] (6 edge orientation channels and a distance threshold of 20). He found that this method outperforms the method without taking orientations into account. We manually optimized the thresholds needed for the edge binarization algorithm (Canny) for each dataset.

Figures 11, 12 and 13 shows the quotient Q_{GM}/Q_{CF} of the quality measure of the two approaches for all frames. Q_{GM} denotes the quality measure for our approach and

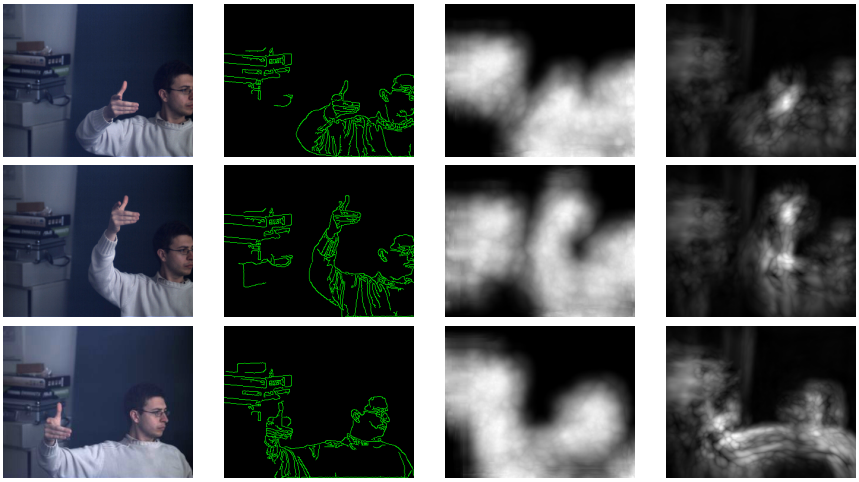

Figure 8: One frame of each of our three datasets: pointing hand (left), open hand (middle), open-close gesture (right). The images in the first row are the originals, the images in the second row are the combined confidence map generated by chamfer matching, and those in the third row are generated by our approach. Notice that with our approach, the maxima in our confidence maps are much more significant. Canny edge image with thresholds 20 and 100 (2nd column), Combined confidence map generated by chamfer based matching(3rd row) and Combined confidence map generated by our approach.

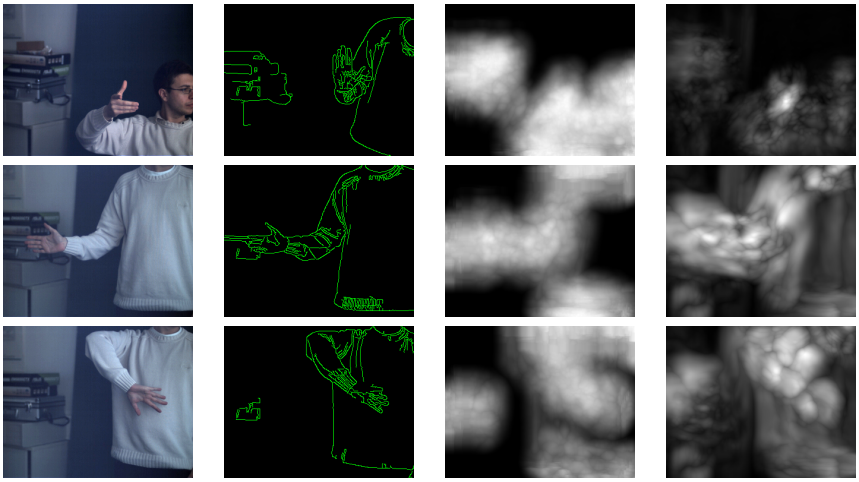

Figure 9: Test data set 2 is an open hand rotating in the image plane Canny thresholds are 20 and 100

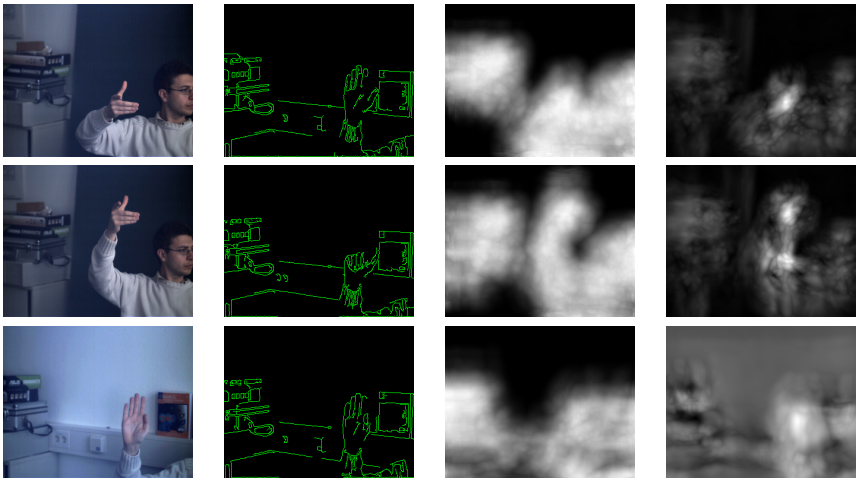

Figure 10: Test data set 3 is an open-closing sequence of the hand. Canny thresholds are 20 and 60

Q_{CF} for the chamfer based approach. Clearly, in most parts of datasets 1 and 3 our approach works better than chamfer based method. Only in the last third of dataset 2, chamfer matching works better. In these frames, none of the templates matches well the orientation of all fingers. Closer inspection suggests that a lot of orientations in these frames happen to be discretized to the right bin in the chamfer based method, which makes it produce a better match with the right template.

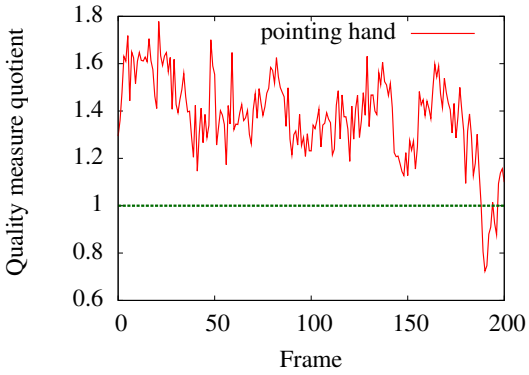

Figure 11: The quotient of the quality between our approach and the chamfer based approach is shown. A value greater than 1 indicates that our approach is better

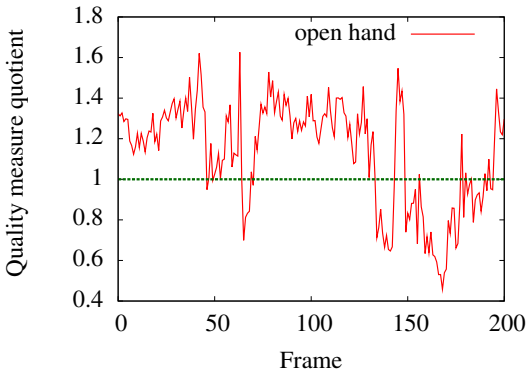

Figure 12: The quotient of the quality between our approach and the chamfer based approach is shown. A value greater than 1 indicates that our approach is better

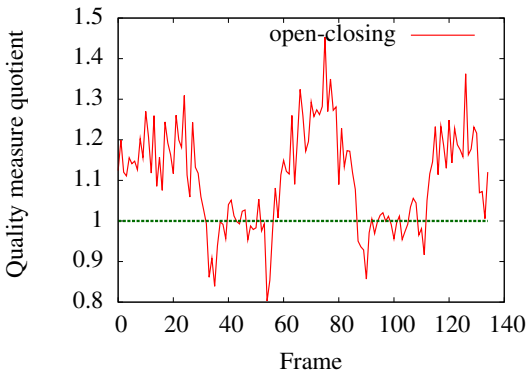

Figure 13: The quotient of the quality between our approach and the chamfer based approach is shown. A value greater than 1 indicates that our approach is better

We measured a frame-rate of about 1.1 fps with our datasets, which comprises the preprocessing of the query images and the convolution with 300 templates. The limiting factor of the computation time of the matching process is the FFT and inverse FFT, which consumes over 90% of the total time.

5 CONCLUSIONS

In this paper, we developed an edge similarity measure for template matching that does not use any thresholds nor discretize edge orientations. Consequently, it works more robustly under various conditions. This is achieved by a continuous edge image similarity measure, which includes a continuous edge orientation distance measure.

In addition, we are able to formulate the edge image distance as a convolution in our method. One major advantage of this is that it lends itself very well for implementation on modern GPUs. Thus, one localization of a template of size 80×80 in an input image of size 320×256 takes only 3 ms.

In about 90% of all images of our test datasets, our method generates confidence maps with fewer maxima that are also more significant. This is better than a state-of-the-art chamfer based method, which uses orientation information as well.

Our edge-based similarity measure also lends itself well to a combination with other features like color or texture, which is straight-forward.

In the future, we plan to test anisotropic kernels for the preprocessing of the templates, which should improve matching quality. Also, we want to apply asymmetric kernel functions to the template image, in order to exploit the knowledge of inner and outer object regions. Furthermore, we will research methods to automatically select the kernel bandwidth parameter.

References

[Athitsos et al., 2004] Athitsos, V., Alon, J., Sclaroff, S., and Kollios, G. (2004). BoostMap: A Method for Efficient Approximate Similarity Rankings. In *IEEE Conference on Computer Vision and Pattern Recognition*.

[Athitsos and Sclaroff, 2001] Athitsos, V. and Sclaroff, S. (2001). 3D Hand Pose Estimation by Finding Appearance-Based Matches in a Large Database of Training Views. In *IEEE Workshop on Cues in Communication*.

[Athitsos and Sclaroff, 2002] Athitsos, V. and Sclaroff, S. (2002). An Appearance-Based Framework for 3D Hand Shape Classification and Camera Viewpoint Estimation. In *IEEE Conference on Automatic Face and Gesture Recognition*.

[Athitsos and Sclaroff, 2003] Athitsos, V. and Sclaroff, S. (2003). Estimating 3D Hand Pose from a Cluttered Image. In *IEEE Conference on Computer Vision and Pattern Recognition*.

[Barrow et al., 1977] Barrow, H. G., Tenenbaum, J. M., Bolles, R. C., and Wolf, H. C. (1977). Parametric Correspondence and Chamfer Matching: Two New Techniques for Image Matching. In *International Joint Conference on Artificial Intelligence*.

[Borgefors, 1988] Borgefors, G. (1988). Hierarchical chamfer matching: A parametric edge matching algorithm. In *IEEE Transaction on Pattern Analysis and Machine Intelligence*.

[Gavrila and Philomin, 1999] Gavrila, D. M. and Philomin, V. (1999). Real-time Object Detection for Smart Vehicles. In *IEEE International Conference on Computer Vision*.

[Huttenlocher et al., 1993] Huttenlocher, D., Klanderman, G. A., and Rucklidge, W. J. (1993). Comparing images using the Hausdorff distance. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[Kato et al., 2006] Kato, M., Chen, Y.-W., and Xu, G. (2006). Articulated Hand Tracking by PCA-ICA Approach. In *International Conference on Automatic Face and Gesture Recognition*.

[Lin et al., 2007] Lin, Z., Davis, L. S., Doermann, D., and DeMenthon, D. (2007). Hierarchical Part-Template Matching for Human Detection and Segmentation. In *IEEE International Conference on Computer Vision*.

[Nvidia, 2008] Nvidia (2008). Compute Unified Device Architecture. `http://www.nvidia.com/object/cuda_home.htm`.

[Olson and Huttenlocher, 1997] Olson, C. F. and Huttenlocher, D. P. (1997). Automatic Target Recognition by Matching Oriented Edge Pixels. In *IEEE Transactions on Image Processing*.

[Shaknarovich et al., 2003] Shaknarovich, G., Viola, P., and Darrell, T. (2003). Fast Pose Estimation with Parameter-Sensitive Hashing. In *IEEE International Conference on Computer Vision*.

[Stenger et al., 2006] Stenger, B., Thayananthan, A., Torr, P. H. S., and Cipolla, R. (2006). Model-Based Hand Tracking Using a Hierarchical Bayesian Filter. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[Stenger, 2004] Stenger, B. D. R. (2004). Model-Based Hand Tracking Using A Hierarchical Bayesian Filter. In *Dissertation submitted to the University of Cambridge*.

[Sudderth et al., 2004] Sudderth, E. B., Mandel, M. I., Freeman, W. T., and Willsky, A. S. (2004). Visual Hand Tracking Using Nonparametric Belief Propagation. In *IEEE CVPR Workshop on Generative Model Based Vision*.

[Thayananthan et al., 2006] Thayananthan, A., Navaratnam, R., Stenger, B., Torr, P., and Cipolla, R. (2006). Multivariate Relevance Vector Machines for Tracking. In *European Conference on Computer Vision*.